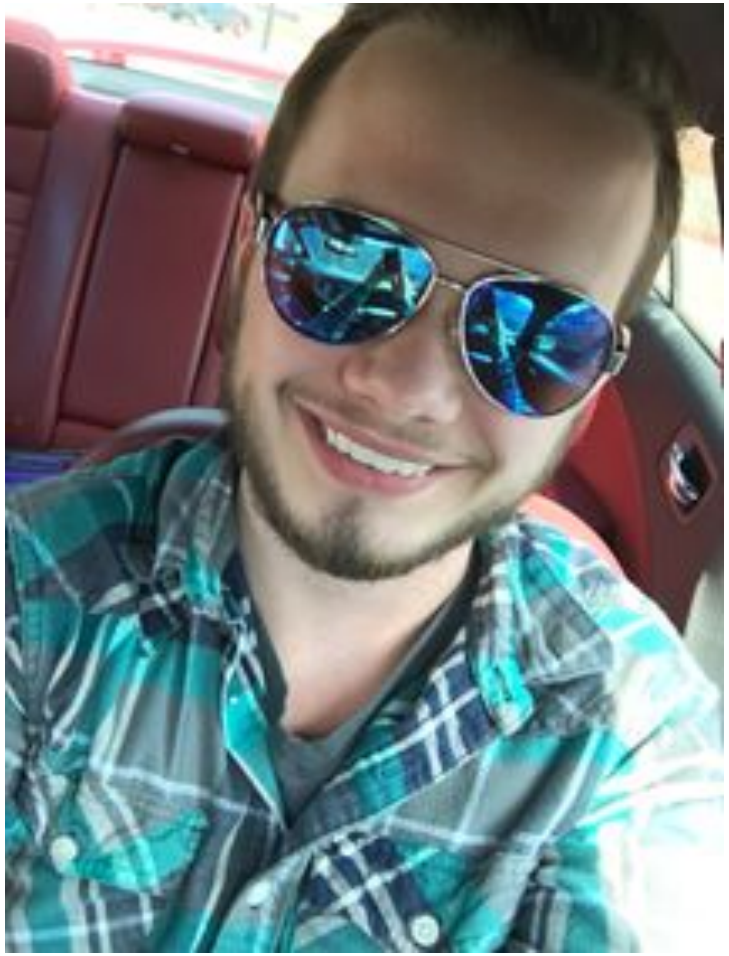
Using OpenCV to Estimate Crowd Size

By: Noah Hickson and Kaleb Burdin

Group 8

Team Members



- Kaleb Burdin (left)
- Noah Hickson (right)

Brief Description

The goal of this project is to accurately estimate the number of people in any given environment. This will enable institutes, such as Arkansas State University, to effectively determine how to enforce safe social distancing to prevent the spread of COVID-19 by being provided with the number of people in one area at any time; this is done by using ThingSpeak, an API used to store and retrieve data. ThingSpeak will display the latest number of people and line chart of all the people counted by the Pi camera. This data will enable institutes to decide where and when social distancing rules should be enforced more strictly or loosely based on the traffic in the area.

Project Activities

Deciding Roles	Decide which team member does each activity	100%
Picking Software	Decide what software is needed for our project	100%
Picking Hardware	Decide what hardware is needed for our project	100%
Test Equipment	Make sure that the hardware and software or up-to-date and compatible with each other	100%
Wire Breadboard	Connect the hardware to the Raspberry Pi and breadboard and ensure that it functions properly	100%
Develop Code	Produce the code needed to make the hardware function correctly and to achieve the desired output of measuring temperature	100%
Test Code	Run the produced code and ensure that there are no errors and that everything functions properly	100%

Gantt Chart

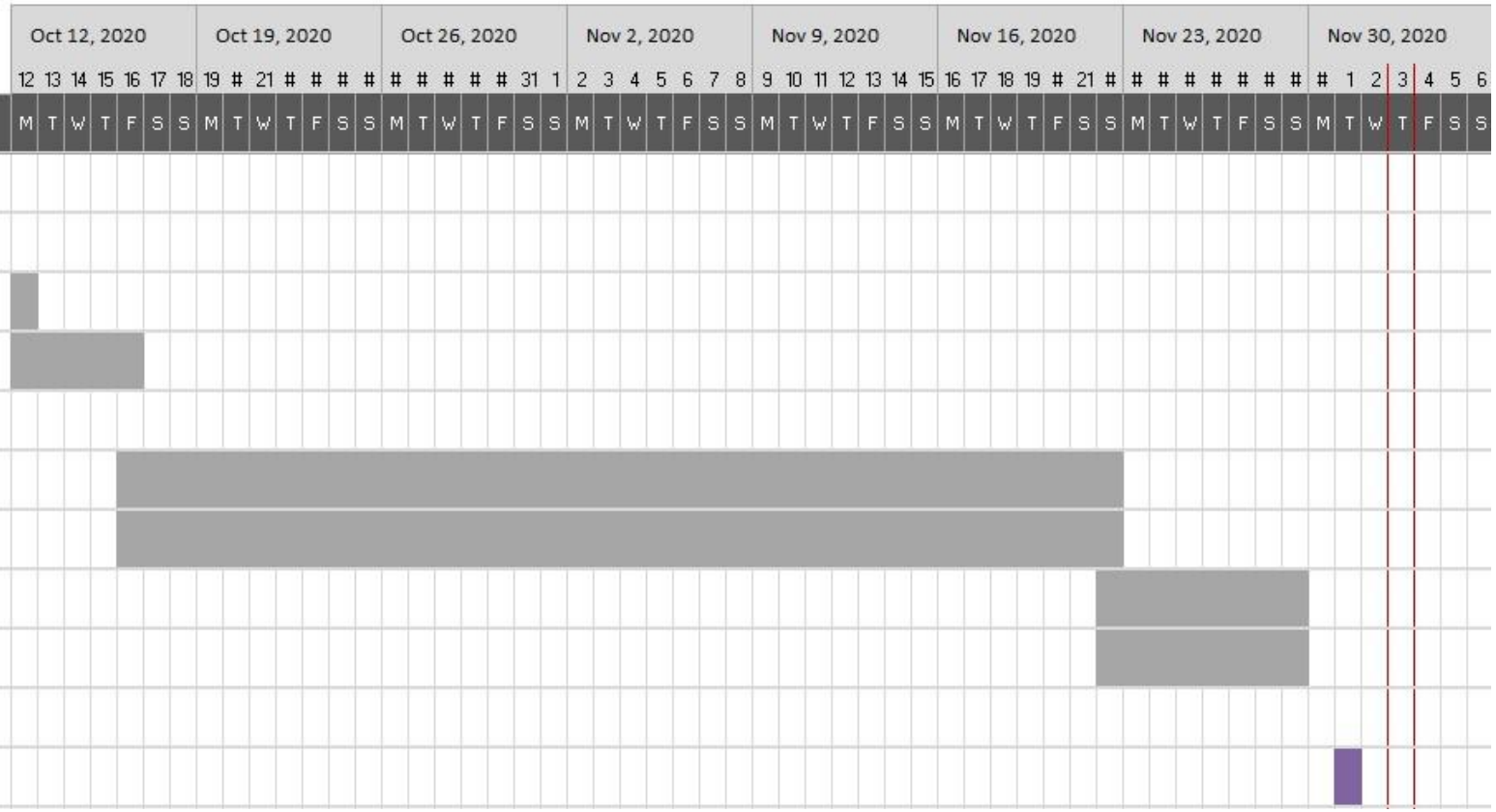
Using OpenCV to Estimate Crowd Size

Noah Hickson
Kaleb Burdin

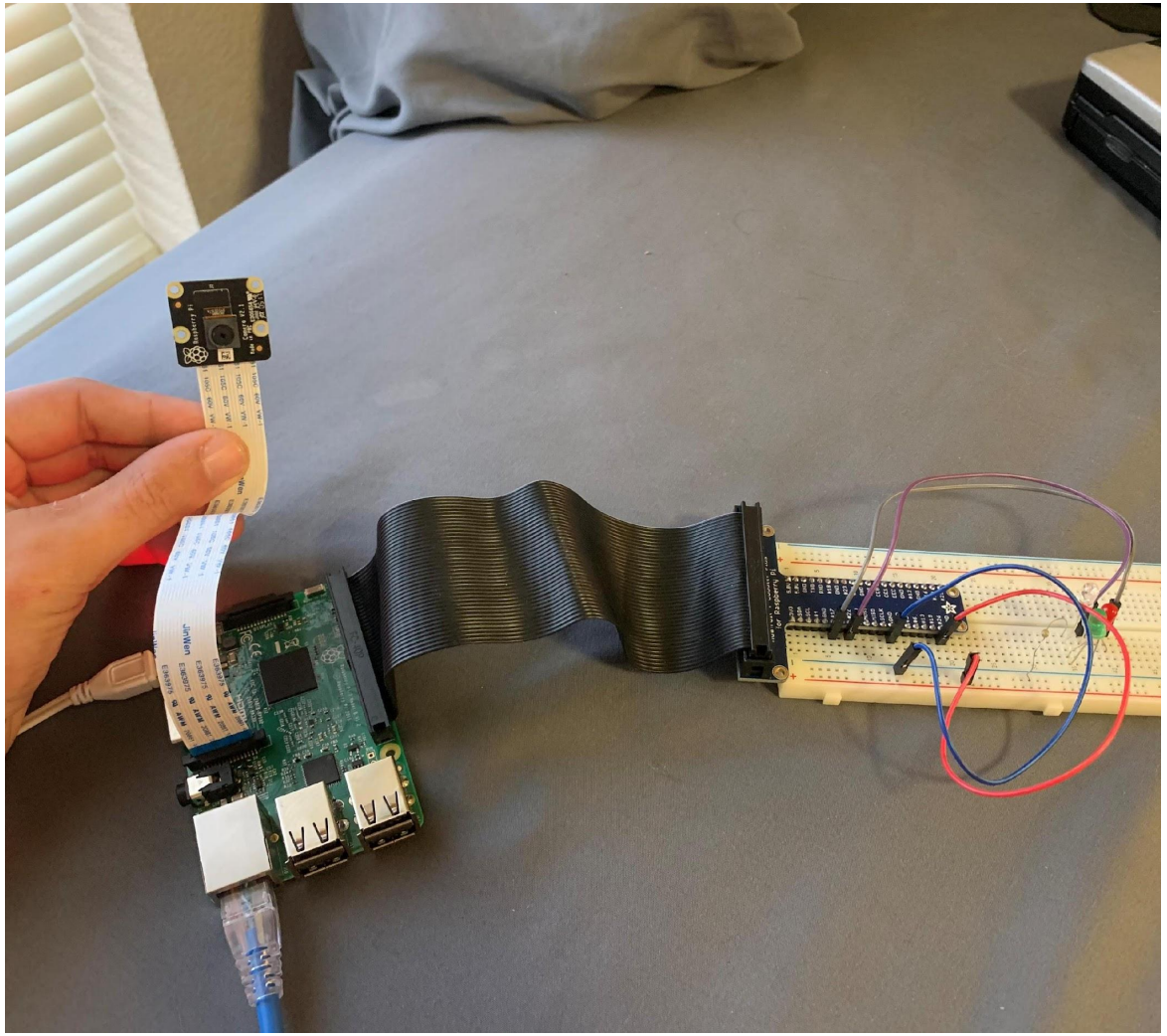
Project Start: Mon, 10/5/2020

Display Week: 2

TASK	ASSIGNED TO	PROGRESS	START	END
Deciding Roles	Noah, Kaleb	100%	10/5/20	10/8/20
Picking Software	Noah, Kaleb	100%	10/8/20	10/10/20
Picking Hardware	Noah, Kaleb	100%	10/10/20	10/12/20
Test Equipment	Noah, Kaleb	100%	10/12/20	10/16/20
Develop Project				
Wire Breadboard	Noah	100%	10/16/20	11/22/20
Develop Code	Kaleb	100%	10/16/20	11/22/20
Test Code	Noah	100%	11/22/20	11/29/20
Debug Code	Kaleb	100%	11/22/20	11/29/20
Present Project				
Present Project	Noah, Kaleb	0%	12/1/20	12/1/20



Hardware Setup



crowd.py

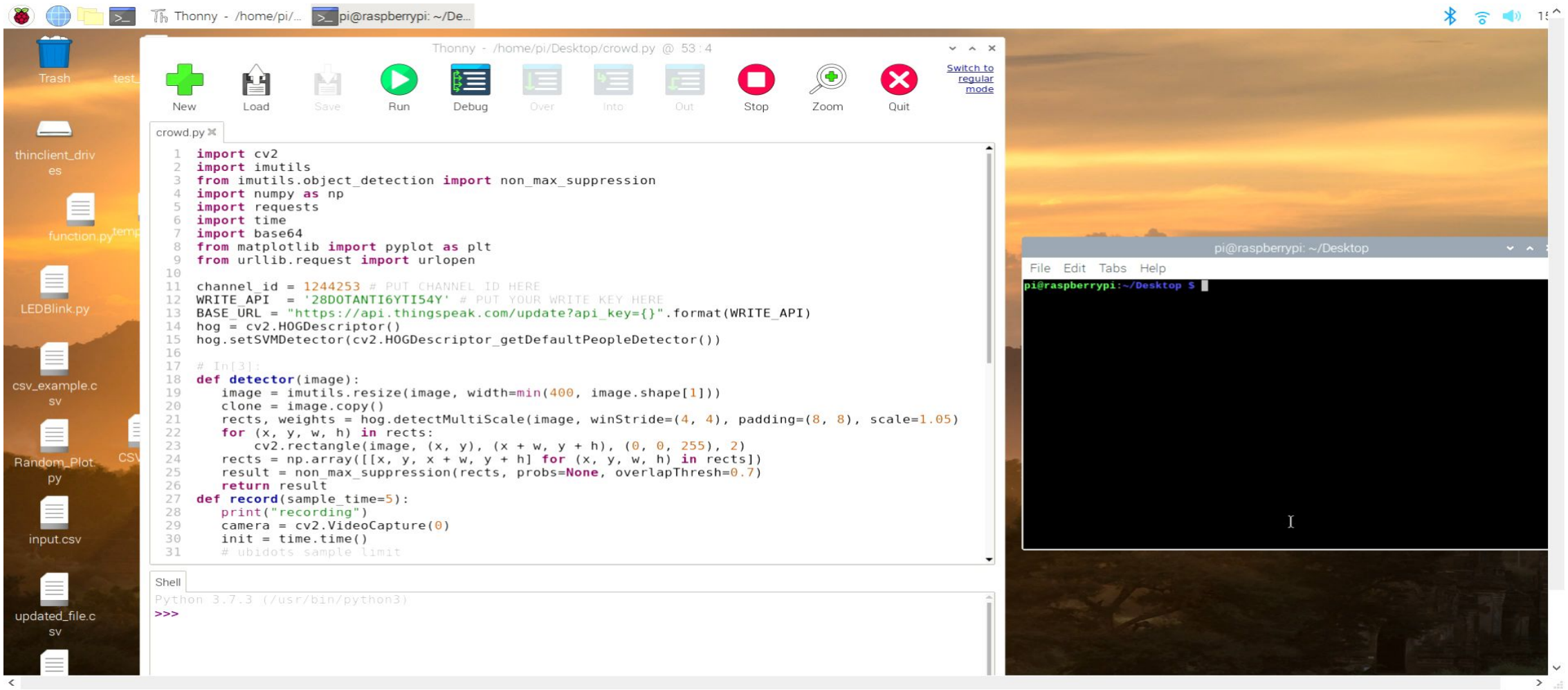
```python
import cv2
import imutils
from imutils.object_detection import non_max_suppression
import numpy as np
import requests
import time
import base64
from matplotlib import pyplot as plt
from urllib.request import urlopen

channel_id = 1244253 # PUT CHANNEL ID HERE
WRITE_API  = '28DOTANTI6YTI54Y' # PUT YOUR WRITE KEY HERE
BASE_URL = "https://api.thingspeak.com/update?api_key={}".format(WRITE_API)
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# In[3]:
def detector(image):
    image = imutils.resize(image, width=min(400, image.shape[1]))
    clone = image.copy()
    rects, weights = hog.detectMultiScale(image, winStride=(4, 4), padding=(8, 8), scale=
    for (x, y, w, h) in rects:
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    result = non_max_suppression(rects, probs=None, overlapThresh=0.7)
    return result
def record(sample_time=5):
    print("recording")
    camera = cv2.VideoCapture(0)
    init = time.time()
    # ubidots sample limit
```

Code

```
    if sample_time < 3:
        sample_time = 1
    while(True):
        print("cap frames")
        ret, frame = camera.read()
        frame = imutils.resize(frame, width=min(400, frame.shape[1]))
        result = detector(frame.copy())
        result1 = len(result)
        print (result1)
        for (xA, yA, xB, yB) in result:
            cv2.rectangle(frame, (xA, yA), (xB, yB), (0, 255, 0), 2)
        plt.imshow(frame)
        plt.show()
        # sends results
        if time.time() - init >= sample_time:
            thingspeakHttp = BASE_URL + "&field1={}".format(result1)
            print(thingspeakHttp)
            conn = urlopen(thingspeakHttp)
            print("sending result")
            init = time.time()
    camera.release()
    cv2.destroyAllWindows()
# In[7]:
def main():
    record()
# In[8]:
if __name__ == '__main__':
    main()
```

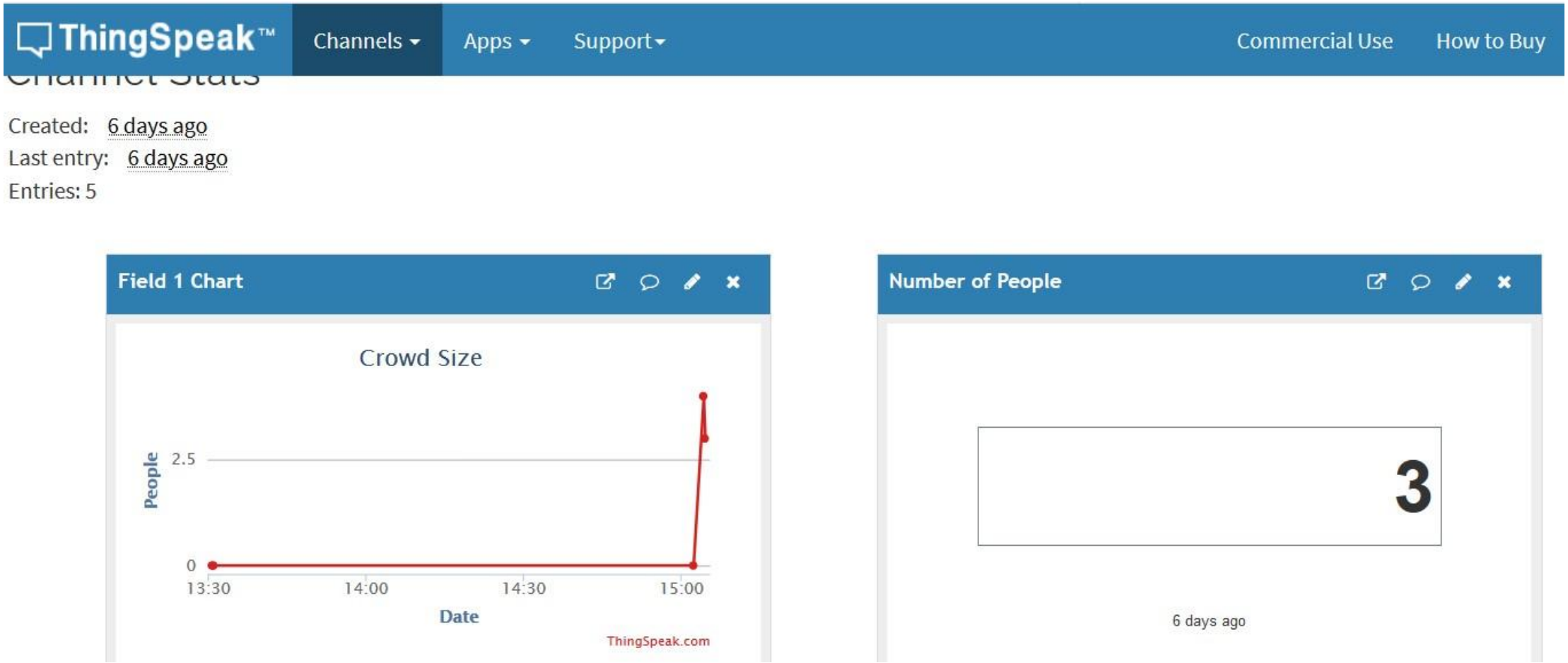
Code Cont.

Video

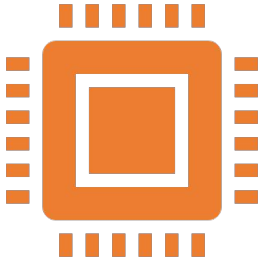

Thonny - /home/pi/...
pi@raspberrypi: ~/De...
Thonny - /home/pi/Desktop/crowd.py @ 53 : 4
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Switch to regular mode
crowd.py
import cv2
import imutils
from imutils.object_detection import non_max_suppression
import numpy as np
import requests
import time
import base64
from matplotlib import pyplot as plt
from urllib.request import urlopen
channel_id = 1244253 # PUT CHANNEL ID HERE
WRITE_API = '28DOTANTI6YTI54Y' # PUT YOUR WRITE KEY HERE
BASE_URL = "https://api.thingspeak.com/update?api_key={}".format(WRITE_API)
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
# In[3]:
def detector(image):
image = imutils.resize(image, width=min(400, image.shape[1]))
clone = image.copy()
rects, weights = hog.detectMultiScale(image, winStride=(4, 4), padding=(8, 8), scale=1.05)
for (x, y, w, h) in rects:
cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)
rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
result = non_max_suppression(rects, probs=None, overlapThresh=0.7)
return result
def record(sample_time=5):
print("recording")
camera = cv2.VideoCapture(0)
init = time.time()
# ubidots sample limit
Shell
Python 3.7.3 (/usr/bin/python3)
>>>
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $
Trash
thinclient_drives
function.py
LEDBlink.py
csv_example.csv
Random_Plot.py
input.csv
updated_file.csv

Picture Used

ThingSpeak Data



Project Setup

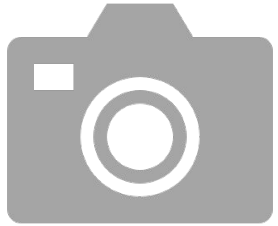


Step 1 – Retrieve the following hardware

Raspberry Pi (any version is fine)

Raspberry Pi Camera

Laptop



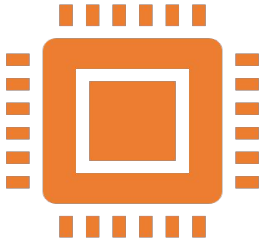
Step 2 – Set up the hardware

Plug in the Raspberry Pi to an outlet to power it on

Lift the plastic clip from the camera port on the Raspberry Pi

Insert the camera ribbon into the camera port and push the clip back into place

Project Setup Cont.

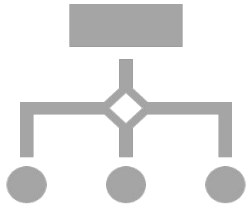


Step 3 – Retrieve the provided python code

Copy the given python code from earlier and paste it into a python IDE

Python IDE must be version 3.7 or later

Make sure you install all Python modules using the command line and pip or python



Step 4 – Set up ThingSpeak

Go to https://thingspeak.com and create an account

Go to channels

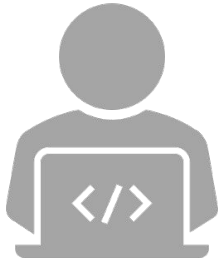
Click ‘New Channel’

Put in your desired information

Go to the python code and set channel_id to the id of the channel you just created

Go to the python code and set write_api to the api of the channel you just created

Project Setup Cont.



Step 4 – Run the program

In the Python IDE, click ‘Run’ then ‘Run Module’

Go to https://thingspeak.com/channels to see your new data

Problems Faced/Facing - 1

Problem 1 (current): Kaleb's Raspberry Pi will no connect to the internet and thus the Raspberry Pi cannot update nor can it pip/python some needed python modules from online

Problem 1 (solution): We are connecting the two cameras to Noah's Raspberry Pi and running the code on his

Problems Faced/Facing - 2

Problem 2 (current): Due to COVID-19, the thermal camera needed to finish the project still has not arrived as shipping has been slowed down across the U.S.

Problem 2 (solution): We are waiting for the camera while thinking of alternative methods to effectively stream IR records

Problems Faced/Facing - 3

Problem 3 (past): We could not get OpenCV to properly install onto Kaleb’s Raspberry Pi

Problem 3 (solution): We had to create an alternative Python version using the command line and delete the current Python version on the Raspberry Pi as it was outdated

Problems Faced/Facing - 4

Problem 4 (past): Kaleb's Pi would not work on an A-State WiFi connection and thus we could not use ThingSpeak

Problem 4 (solution): The Pi still did not work after adding the device to the Resnet Portal so we've been using a private network at our houses.